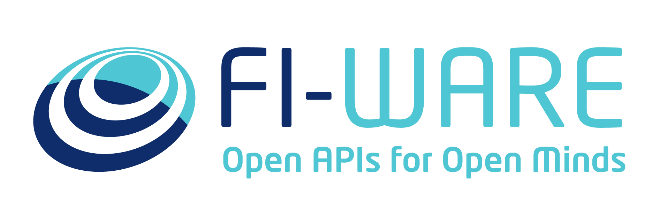
**MNCA IOT SmartCity Technical Architecture**

****

****

**DSI Métropole Nice Côte d’Azur**

**IOT & Smart City Platform**

|  |  |
| --- | --- |
| VERSION: draft | REVISION DATE: 02/11/2017 |

Approval of the Technical Architecture indicates an understanding of the purpose and content described in this deliverable. By signing this deliverable, each individual agrees with the content contained in this deliverable.

|  |  |  |  |
| --- | --- | --- | --- |
| **Approver Name** | **Title** | **Signature** | **Date** |
| Stéphane Roux | ICT Quality Assurance Engineer |  |  |
| Lionel Chaudanson | Responsable technique, IOT & Smart City |  |  |
| Serge Massiera | Directeur des Systèmes d'Information |  |  |
|  |  |  |  |
|  |  |  |  |

[1 Section 1 DOCUMENT SCOPE 6](#_Toc499654965)

[1.1 Context 6](#_Toc499654966)

[1.2 Glossary 8](#_Toc499654967)

[2 Section 2 OVERALL TECHNICAL ARCHITECTURE 9](#_Toc499654968)

[2.1 System Architecture Context Diagram 9](#_Toc499654969)

[2.2 System Architecture Model 13](#_Toc499654970)

[2.2.1 Overall Architectural Considerations 14](#_Toc499654971)

[2.3 System Architecture Component Definitions 17](#_Toc499654976)

[2.3.1 Docker Images Management Components 17](#_Toc499654977)

[2.3.2 Docker Swarm Management Component 18](#_Toc499654978)

[2.3.3 Security Component IDM GE 18](#_Toc499654979)

[2.3.4 Orion Context Broker Component 19](#_Toc499654980)

[2.3.5 MongoDB Cluster Component 19](#_Toc499654981)

[2.3.6 Citus/PostgreSQL Cluster Component 20](#_Toc499654982)

[2.3.7 Galera/MySQL Cluster Component 20](#_Toc499654983)

[2.3.8 Cosmos Big Data Component 20](#_Toc499654984)

[2.3.9 Knowage Analytics Component 21](#_Toc499654985)

[2.3.10 CKAN Open Data Component 21](#_Toc499654986)

[2.3.11 IDAS IOT Component 22](#_Toc499654987)

[2.3.12 CEP Component 23](#_Toc499654988)

[2.3.13 Application Mashup Components 23](#_Toc499654989)

[2.3.14 Business Ecosystem Component 24](#_Toc499654990)

[2.3.15 Development Components 24](#_Toc499654991)

[2.3.16 Monitoring Components 25](#_Toc499654992)

[2.3.17 Poi Data Provider Component 25](#_Toc499654993)

[2.3.18 GIS Data Provider Component 26](#_Toc499654994)

[3 Section 3 SYSTEM ARCHITECTURE DESIGN 26](#_Toc499654995)

[3.1 Docker Images Management Component 26](#_Toc499654996)

[3.1.1 Component Functions 26](#_Toc499654997)

[3.1.2 Technical Considerations 26](#_Toc499654998)

[3.1.3 Selected Product(s) 27](#_Toc499654999)

[3.1.4 Selection Rationale 27](#_Toc499655000)

[3.1.5 Architecture Risks 27](#_Toc499655001)

[3.2 Docker Swarm Manager Component 27](#_Toc499655002)

[3.2.1 Component Functions 27](#_Toc499655003)

[3.2.2 Technical Considerations 27](#_Toc499655004)

[3.2.3 Selected Product(s) 27](#_Toc499655005)

[3.2.4 Selection Rationale 27](#_Toc499655006)

[3.2.5 Architecture Risks 27](#_Toc499655007)

[3.3 Security Component IDM GE 27](#_Toc499655008)

[3.3.1 Component Functions 27](#_Toc499655009)

[3.3.2 Technical Considerations 29](#_Toc499655010)

[3.3.3 Selected Product(s) 30](#_Toc499655011)

[3.3.4 Selection Rationale 30](#_Toc499655012)

[3.3.5 Architecture Risks 30](#_Toc499655013)

[3.4 Orion Context Broker Component 30](#_Toc499655014)

[3.4.1 Component Functions 30](#_Toc499655015)

[3.4.2 Technical Considerations 30](#_Toc499655016)

[3.4.3 Selected Product(s) 31](#_Toc499655017)

[3.4.4 Selection Rationale 31](#_Toc499655018)

[3.4.5 Architecture Risks 31](#_Toc499655019)

[3.5 MongoDB Component 31](#_Toc499655020)

[3.5.1 Component Functions 31](#_Toc499655021)

[3.5.2 Technical Considerations 31](#_Toc499655022)

[3.5.3 Selected Product(s) 31](#_Toc499655023)

[3.5.4 Selection Rationale 31](#_Toc499655024)

[3.5.5 Architecture Risks 31](#_Toc499655025)

[3.6 Citus/PostgreSQL cluster Component 32](#_Toc499655026)

[3.6.1 Component Functions 32](#_Toc499655027)

[3.6.2 Technical Considerations 32](#_Toc499655028)

[3.6.3 Selected Product(s) 32](#_Toc499655029)

[3.6.4 Selection Rationale 32](#_Toc499655030)

[3.6.5 Architecture Risks 32](#_Toc499655031)

[3.7 Galera/MySQL cluster Component 32](#_Toc499655032)

[3.7.1 Component Functions 32](#_Toc499655033)

[3.7.2 Technical Considerations 32](#_Toc499655034)

[3.7.3 Selected Product(s) 32](#_Toc499655035)

[3.7.4 Selection Rationale 32](#_Toc499655036)

[3.7.5 Architecture Risks 32](#_Toc499655037)

[3.8 Cosmos Big Data Component with Cygnus & Comet 32](#_Toc499655038)

[3.8.1 Component Functions 32](#_Toc499655039)

[3.8.2 Technical Considerations 33](#_Toc499655040)

[3.8.3 Selected Product(s) 33](#_Toc499655041)

[3.8.4 Selection Rationale 33](#_Toc499655042)

[3.8.5 Architecture Risks 33](#_Toc499655043)

[3.9 Knowage Analytics Component 33](#_Toc499655044)

[3.9.1 Component Functions 33](#_Toc499655045)

[3.9.2 Technical Considerations 33](#_Toc499655046)

[3.9.3 Selected Product(s) 33](#_Toc499655047)

[3.9.4 Selection Rationale 33](#_Toc499655048)

[3.9.5 Architecture Risks 33](#_Toc499655049)

[3.10 CKAN Open Data Component 33](#_Toc499655050)

[3.10.1 Component Functions 33](#_Toc499655051)

[3.10.2 Technical Considerations 33](#_Toc499655052)

[3.10.3 Selected Product(s) 33](#_Toc499655053)

[3.10.4 Selection Rationale 33](#_Toc499655054)

[3.10.5 Architecture Risks 33](#_Toc499655055)

[3.11 IDAS IOT Component 33](#_Toc499655056)

[3.11.1 Component Functions 34](#_Toc499655057)

[3.11.2 Technical Considerations 34](#_Toc499655058)

[3.11.3 Selected Product(s) 34](#_Toc499655059)

[3.11.4 Selection Rationale 34](#_Toc499655060)

[3.11.5 Architecture Risks 34](#_Toc499655061)

[3.12 CEP Component 34](#_Toc499655062)

[3.12.1 Component Functions 34](#_Toc499655063)

[3.12.2 Technical Considerations 34](#_Toc499655064)

[3.12.3 Selected Product(s) 34](#_Toc499655065)

[3.12.4 Selection Rationale 34](#_Toc499655066)

[3.12.5 Architecture Risks 34](#_Toc499655067)

[3.13 Application Mashup Components 34](#_Toc499655068)

[3.13.1 Component Functions 34](#_Toc499655069)

[3.13.2 Technical Considerations 34](#_Toc499655070)

[3.13.3 Selected Product(s) 34](#_Toc499655071)

[3.13.4 Selection Rationale 34](#_Toc499655072)

[3.13.5 Architecture Risks 34](#_Toc499655073)

[3.14 Business Ecosystem Component 34](#_Toc499655074)

[3.14.1 Component Functions 35](#_Toc499655075)

[3.14.2 Technical Considerations 35](#_Toc499655076)

[3.14.3 Selected Product(s) 35](#_Toc499655077)

[3.14.4 Selection Rationale 35](#_Toc499655078)

[3.14.5 Architecture Risks 35](#_Toc499655079)

[3.15 Development Components 36](#_Toc499655080)

[3.15.1 Component Functions 36](#_Toc499655081)

[3.15.2 Technical Considerations 36](#_Toc499655082)

[3.15.3 Selected Product(s) 36](#_Toc499655083)

[3.15.4 Selection Rationale 36](#_Toc499655084)

[3.15.5 Architecture Risks 36](#_Toc499655085)

[3.16 Monitoring Components 36](#_Toc499655086)

[3.16.1 Component Functions 36](#_Toc499655087)

[3.16.2 Technical Considerations 36](#_Toc499655088)

[3.16.3 Selected Product(s) 36](#_Toc499655089)

[3.16.4 Selection Rationale 36](#_Toc499655090)

[3.16.5 Architecture Risks 36](#_Toc499655091)

[4 Section 4 System Construction Environment 36](#_Toc499655092)

[4.1 Preliminary test Environment 36](#_Toc499655093)

[4.2 Development Environment 37](#_Toc499655094)

[4.2.1 Developer Workstation Configuration 37](#_Toc499655095)

[4.2.2 Supporting Development Infrastructure Configuration 37](#_Toc499655096)

[4.3 QA Environment 37](#_Toc499655097)

[4.3.1 QA environment Configuration 37](#_Toc499655098)

[4.3.2 Supporting QA Infrastructure Configuration 37](#_Toc499655099)

[4.4 Acceptance Environment 37](#_Toc499655100)

[4.4.1 Acceptance environment Configuration 37](#_Toc499655101)

[4.4.2 Supporting Acceptance Infrastructure Configuration 37](#_Toc499655102)

# Section 1 DOCUMENT SCOPE

## Context

***Document Scope*** *describes the context and the goals of this document in a narrative.*

This document describes the Technical Architecture of the IOT Smart City Platform that:

* satisfies business requirements as documented in the roadmap (ref to provide),
* satisfies technical, operational and transitional requirements.

New technologies, particularly digital technologies, allow the development of new services and new uses for improving the quality of life in the respect the environment in today’s metropolis.

With the advent of IOT, Big data and AI, the concept of smart city can come to reality.

To anticipate these paradigm shifts, the metropolis of Nice launched in 2008 under the leadership of Mr Estrosi, an initiative multichannel scale that aims to achieve the digital transformation of the territory in the PACA region.

The main idea behind this major initiative lies in the establishment of complex synergies between local authorities, the university (Research and education) and economic actors (industry and start-ups).

These synergies are intended to develop the best technologies and uses them more successful in the service to citizens.

Indeed, advanced technologies and the use intelligent interest data should allow today, to improve the quality of living, well-being and the functioning of the urban unit in the constraints of sustainable development.

The challenge for the Nice metropolis is also to maintain control of these while promoting economic spin-offs and job creation.

The recent obtaining of the label IDEX (January 2016) ideally positions the metropolis of Nice and the Côte d'Azur University (UCA) in the national pole of excellence and therefore European.

The IDEX and the Nice metropolis have chosen to strengthen, in particular three complementary axes: smart-city, health / silver economy and telco / electronics.

These cutting-edge technological and societal initiatives are part of an organization

rationality of the territory:

* The plain of Var for Eco-Valley / smart-city
* Pasteur area for health / silver economy
* Sophia-Antipolis for electronics / Telco / Pharmaceutical /

The coherence and functionality of the whole are strategically driven by the public transport network including the T2 and T3 tram lines which will connect directly to the airport and train station, eco-valley and Arenas stadium.

Local and university authorities have therefore established a single framework facilitating the functional translation of the most advanced concepts from laboratories to the end user. Small and large businesses have all the keys to development of innovations and their economic projects.

In order, to make this transformation into a smart city, a reality, Nice Metropolis after a period of POC, have decided to reach a new level.

The main objectives for this transformation are among:

* Provide a coherent and efficient framework for application development
* App store and Market Place
* Open data
* Labeling of applications
* Better visibility for users
* Aggregate the different sensor subnetworks into a metropolitan collection backbone network
* Adopt a logic of optimization and rationalization
* Capitalize on existing networks managed by the ISD (470 km of fiber optics, radio bridges, VPN-M2M)
* Standardize and store information to facilitate collaboration between operational departments
* Share and enrich these data by experimenting with industrialists, laboratories, researchers, communities, citizens, ...
* Update continuously various predictions by integrating real-time data
* Develop an hyper-vision system for the management of exceptional events (floods, demonstrations, disorders, ...)

Therefore, after a study conducted by Eridanis on the state of art of IOT platforms for Smart City, a roadmap has been provided for the setup of such a system, based mainly on Fiware.

In the referenced roadmap document, the needs have been described through the following global schema:

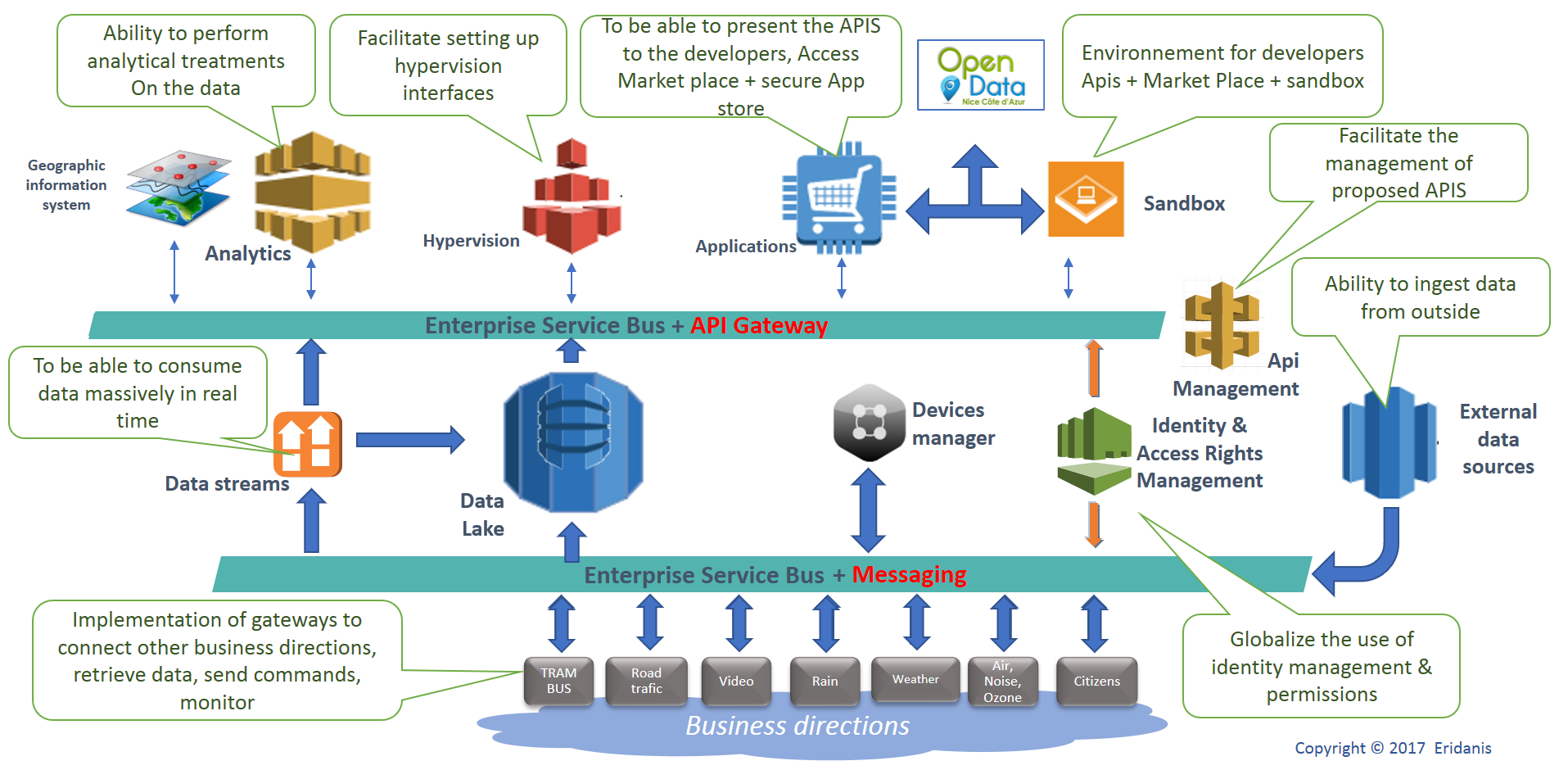


Figure 1: Target Logical Smart City Platform

The goal of this Technical Architecture is to define the technologies, products, and techniques necessary to install and support the system, and to ensure that the system components are compatible and comply with the enterprise-wide standards and direction defined by Nice Côte d’Azur metropolis (hereafter referred as MNCA).

This document will also:

* Identify and explain the risks inherent in this Technical Architecture;
* Define baseline sizing, archiving and performance requirements;
* Identify the hardware and software specifications for the Development, Testing, QA and Production environments;
* Identify monitoring and support tools for maintenance in operational condition
* Define procedures for data migration among the environments (current data warehouse).

## Glossary

**SOA:** Service Oriented Architecture, is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.

**Micro Services Architecture**: Microservices is a variant of the Service-Oriented Architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight, it naturally enforces a modular structure.

**FIWARE**: is an independent open community whose members are committed to materialize the FIWARE mission, that is: “to build an open sustainable ecosystem around public, royalty-free and implementation-driven software platform standards that will ease the development of new Smart Applications in multiple sectors”.

**Fiware platform**: is a platform providing a simple but powerful set of APIs that ease the development of Smart Applications in multiple vertical sectors: <https://www.fiware.org/>, it is based mainly on a Micro Services Architecture, where components expose RESTful APIs.

**IOT**: Internet Of Things

**OMA**: [Open Mobile Alliance](http://www.openmobilealliance.org/) : OMA is a non-profit organization that delivers open specifications for creating interoperable services that work across all geographical boundaries, on any bearer network.

**TM Forum**: [TM Forum](https://www.tmforum.org/) is the global industry association that drives digital transformation of the communications industry through collaboration.

**GE**: Generic Enabler, Fiware used terminology to design an essential module of the platform

**API**: Application Programming Interface

**RESTful API**: also referred to as a RESTful web service, it is based on representational state transfer ([REST](https://en.wikipedia.org/wiki/Representational_state_transfer)) technology, an architectural style and approach to communications often used in web services development as an alternative to [SOAP](https://en.wikipedia.org/wiki/SOAP).

**Context Broker**: According to Gartner:

“A context broker is a service that is designed to gather reachable context data of a variety of types, sources and velocity. It then applies conditioning, integration, rules and analytics to derive the reduced prepared context data, actionable at a point of business decision by a system or a human.”

Functionally speaking, context enables the data to be empowered by the interaction with other pieces of data. In a nutshell it can be consider as a data exchange bus, providing access to context information through publish/subscribe paradigm (this is kind of “MQTT” for context data, through RESTful API).

**NGSI**: New Generation Services Interface, [OMA referenced API specifications](http://www.openmobilealliance.org/release/NGSI/) defined ([NGSI V9 and V10](https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI-9/NGSI-10_information_model)) and used with amendments by Fiware, for use through Context Broker, as the dorsal for the platform: NGSI V1 and now [NGSI V2](http://fiware.github.io/context.Orion/api/v2/stable/) which now the reference api version.

**Docker**: is a virtualization software technology providing containers to run software applications in isolation: [Docker site](https://www.docker.com/), [Docker Documentation](https://docs.docker.com/)

**Swarm**: [Docker Swarm](https://docs.docker.com/engine/swarm/) provides native clustering functionality for Docker containers, which turns a group of Docker engines into a single, virtual Docker engine.

**VRF**: Virtual Routing and Forwarding network component.

**GDPR**: General Data Protection Regulation

**RIA**: Rich Internet Application

**UI**: User Interface

**Ontology**: formal specification of a conceptualization, used to explicitly capture the semantics of a certain reality

# Section 2 OVERALL TECHNICAL ARCHITECTURE

## 2.1 System Architecture Context Diagram

The **Smart city plarform Architecture Context Diagram** provides the “big picture” view of the global system’s architecture, and puts it in context with the rest of the Performing Organization’s systems portfolio, illustrating how the system’s hardware and software platforms fit into the existing environment.

The MNCA IOT Smart City Platform will be deployed upon the 3 data centers owned and managed by the MNCA DSI

* The main data center is Phoenix,
* The secondary is Bosio (currently underworks for revamping for 6 months),
* The third one (the smallest) is Biscarra, (the configuration has to be reconsidered but after Bosio .
* The hardware infrastructure is based on bare metal servers (Cisco and a few remaining HP)
* The servers are blades in enclosures, 3 enclosures per cabinet
* The machines are managed with vSphere and VMWare ESX as virtualization layer, vCenter, vMotion.
* There is a redundant SAN (EMC), dispatched in main and secondary DC. (capacity ?)
* It exists also a redundant NAS system for low I/O file storage.

MNCA Network is managed with a cluster of firewall/routers, managing the separation in 4 different virtual networks

* Public DMZ for applications
* Private DMZ
* LAN
* Field VLANs for IOT gateways, devices and sensors

There’s several Load balancing F5 Big IP appliances at strategic points to distribute load to data centres.



Figure 2: MNCA Infrastructure Global View

The previous schema shows globally the IT and network infrastructure in place for MNCA.

Since servers are dispatched through the 4 VLANs independently from the cabinets, the schema is not the true image of cabinets repartition in the different VLANs

The existing IT services for MNCA are deployed through this available infrastructure.

The Smart City platform will as well be deployed on the same data center infrastructure, but a new DMZ network will be created to host most of the Smart City platform components.

IOT components such as IOT agents, will be deployed under the IOT field dedicated VLANs (industrial VRF).

Nevertheless, some new hardware infrastructure will most probably be required, to provide network, computing power and storage.

The envisaged interaction between those services and the smart City platform is for the moment limited to:

* use MNCA mail services (outgoing/incoming)
* connect existing data sources to the new system instead of the current used data warehouse
* use global MNCA security system based on Microsoft Adam (LDAP server), when possible
* the platform will interface to several industrial networks to gather information from field such as: road traffic, water management, sanitation network, environmental monitoring (noise, air quality, ozone), public transport (buses, tram), weather, video monitoring, citizens, buildings monitoring...

After a selection process it has been agreed that the MNCA smart city platform will be mainly based on [FIWARE](https://www.fiware.org/)

The Fiware community propose a set of modules called Generic Enablers, which provide the essential software bricks to set a complete smart city platform, like a kind of “Lego”.

The FIWARE architecture is by nature open, and based on open source modules available in a [dedicated catalog](https://catalogue.fiware.org/)

The most important to mention is that FIWARE has deliberately chosen to go towards a container based architecture, this limits the dependency with Openstack initially thought as the preferred cloud stack for deploying Fiware platforms using virtual machines.

Fiware started in 2014, which was just the beginning of the “Dockerization” of the server software world, the Fiware consortium couldn’t stay without going towards container based deployment, Docker is now an officially supported way to deploy Fiware components in a very clear and easy way.

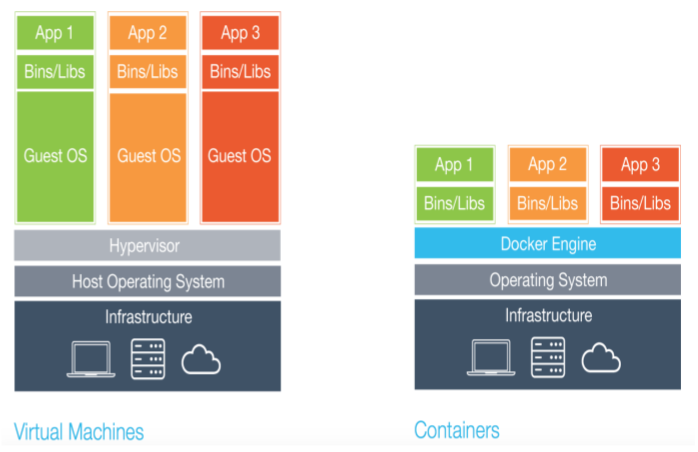


Figure 3: Docker compared to Virtual Machines

MNCA have the opportunity to introduce new way to use its infrastructure using container based architecture, by choosing to use Docker to deploy its own Fiware plarform as a reference Smart City Platform.

The global architecture of the MNCA Fiware platform will rely on a Docker Swarm composed of nodes dispatched in the different VLANs.

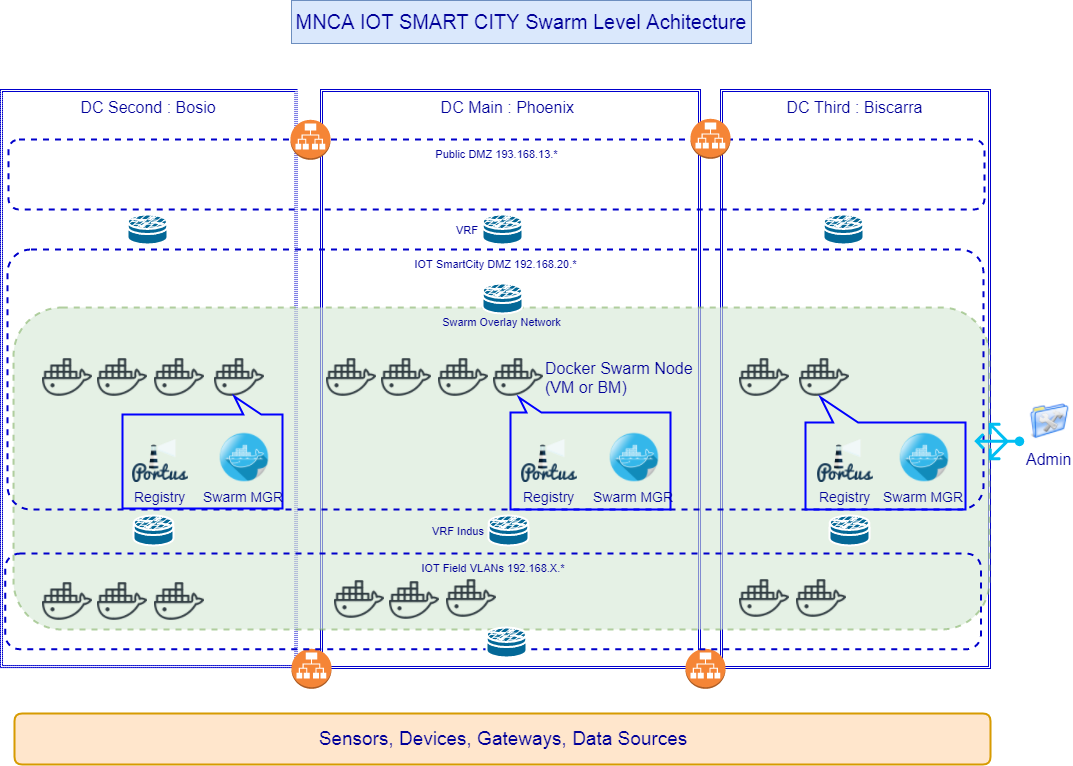


Figure 4: Docker swarm deployment for MNCA Smart city Fiware platform

A Docker swarm (Cluster management integrated with Docker Engine) solution has been chosen in a first step because of the ease of use and configuration.

The learning curve of Docker container orchestration tool: Swarm, is less sloppy than other solution like Kubernetes (the open source declination of the Google internal orchestration tool: Borg), which sit at the top in term of features, openness, and adoption, among cloud orchestration tools today’s landscape.

Even it is the right choice to start with simple Docker Swarm, it will be easy to add later another orchestration tool like Kubernetes, since it will be proposed as the Swarm complement in Docker Enterprise distribution ([very recent news](https://thenewstack.io/docker-fully-embraces-kubernetes/)), the upgrade feasibility is guaranteed.

Then the MNCA FIWARE smart city platform will be deployed in a global docker swarm covering the 3 data centers.

The repartition of the traffic between the data centers will be managed by F5 BigIP load balancing appliance.

However, to minimize configuration task on load balancer (limit to load repartition between data centers of main categories of services), for the lower granularity of applications, and internal components of the platform, simple load balancing and fail-over mechanisms are managed automatically by the Docker Swarm (number of instances of a service).

For the deployment of the platform over a Swarm, we still need to provision virtual machines or bare metal servers and add it to the swarm.

For the moment VM are created using VMWare vSphere, it’s simple, but it comes with a cost per each created VM…

VMware proposes vSphere® Integrated Containers, which allows to manage containers from vSphere, but it doesn’t change the cost for the deployed VMs a license fee is still applied.

A solution is proposed: [VMWARE INTEGRATED OPENSTACK](https://www.vmware.com/fr/products/openstack.html), (VIO), which allows to leverage OpenStack from VMWare vSphere, the infrastructure is managed using OpenStack tools, but there’s still license fee to pay per CPU (VIO 4.0 per-CPU license and Support and Subscriptions (SnS)).

Despite this license fees issue, VIO offers Container execution on OpenStack: for customers seeking to modernize their traditional applications or to design new ones, based on containers, VMware Integrated OpenStack offers **Kubernetes** support ready for production with pooling and persistent storage (volumes). This solution ready for use for who want to run containers on OpenStack and exploit VMware's virtualization technologies.

The choice to use the current way or VIO with vSphere, to manage the platform infrastructure need to be assessed with induced costs.

Anyway, using only pure OpenStack tools to manage the platform infrastructure is a solution to consider at least to lower TCO in terms of license fees.

## System Architecture Model

The **System Architecture Model** represents the various architecture components that comprise the system, and shows the important interrelationships.

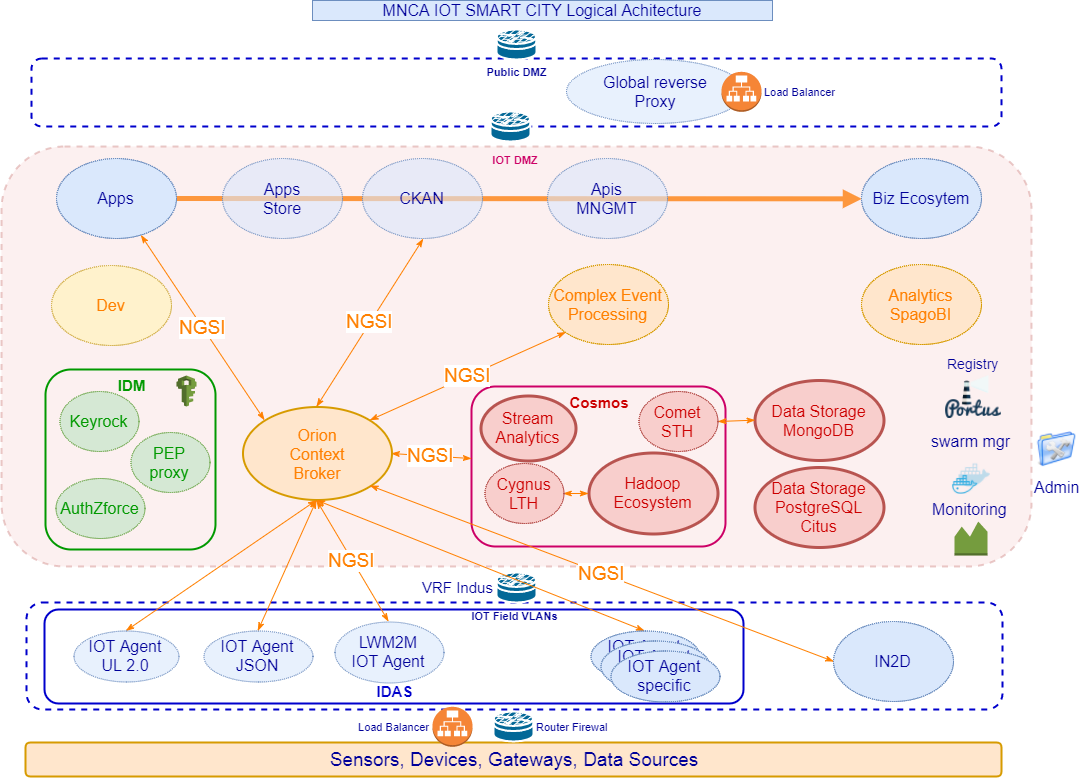


Figure 5: MNCA IOT Smart City Platform logical architecture

* Not all the relations between components have been represented in the above schema to preserve readability.
* Most of the components provide interfaces through RESTful API over HTTP/HTTPS, so that the platform is SOA (Service Oriented Architecture), and micro services based.
* The central component that provides the main “context data exchange bus” is the Orion Context Broker, this is the Fiware Generic Enabler (GE) to consider as the cornerstone of the platform.
* As any interaction with this component is done using NGSI RESTful APIs, this is the main communication stream in the platform, that is represented on the above figure.
* As well the transversal arrow to Biz ecosystem GE, represent the link of each component to it for platform assets catalog management and monetization.
* Are not represented the relations to the identity manager (IDM GE) component, which are generally used to authorize access to the component through a proxy to identity manager.
* The aim with the IDM (Identity Management) component, is to try to offer a centralized SSO management over the whole platform, allowing the central control of access authorization to platform components.
* The data storage component is not a Fiware component, this should be the SQL data tank of any Fiware module using SQL database, in case it’s impossible to change (a lot of Fiware modules rely on MySQL database) to PostgreSQL, we can use MariaDB.
* The general idea with the data storage components in the representation is to emphasis the fact that all the databases used by Fiware modules should be centralized in some clusters based on needed database server, which is true for SQL database and NoSQL database, so that relation links have not been represented on the above schema
* Inside the DEV component on the above figure are the Paas and the tools needed to develop applications that will use the Fiware ecosystem, this encompass development tools but also QA tools for testing and validation purpose, which can have specific needs in term of access to data. Most probably for development and QA, it will be required to deploy some Fiware components or even a complete platform for staging.
* A global reverse proxy and load balancer (F5 BigIP) will be the main entry point to the smart city platform.

### 2.2.1 Overall Architectural Considerations

*The* ***Overall Architectural Considerations*** *section defines how additional technical requirements have been addressed by the architecture. items in this section include:*

* *Deployment strategy*
* *Security Strategy*
* *Availability requirements*
* *Performance requirements*
* *Accessibility*
* *Database management*
* *Transaction volumes*
* *Concurrent users & devices*
* *Data import and export*
* *Disaster recovery*
* *GDPR*

#### Deployment Strategy

As stated previously, the MNCA Smart City platform will be deployed using Docker containers in a global swarm (group of Docker engines).

In this first step Docker engines will be deployed on Virtual machines dispatched on the 3 data centers (when they will be available, for the first step installation the platform will be deployed on the main data center).

The different modules will be grouped in services and stacks (Docker concepts).

Only Database servers, will be deployed in clusters, at disposal for use by other components of the platform.



#### Security Strategy

Application security is achieved through a global identity and access rights management component the Fiware Identity Management GE, providing SSO and secure access to other components of the platform, according to requester authorizations.

The identity management component will be connected to the MNCA LDAP (MS ADAM based).

By using [Docker Swarm's secrets](https://blog.docker.com/2017/02/docker-secrets-management/), component’s secrets (passwords, tokens, SSH private keys, certificates) can be safely stored and used in services composition files without compromising security.

Docker Images used for the platform will be locally managed in a secured registry.

#### Availability requirements

All the components used in the platform will be deployed in a way providing high availability.

The services and the associated databases should ideally be deployed on the 3 data centers,

The third data center remaining off-line most of time, should host replicas of the databases, updated in real-time.

The use of Docker provides an easy way to deploy highly available services, by deploying several containers on several nodes of a swarm, using docker compose files or docker stack files.

#### Performance requirements

Any of the components will be scalable by adding some instance of containers to scale-up the service.

Through the use of Docker it is very easy to achieve by changing services configuration on the fly.

#### Accessibility

Nothing specific is included in Fiware for accessibility, this item is to be considered at application UI level, most probably using accessibility features from operating systems or internet browsers that host the client application.

#### Database management

Different kind of databases are used among the Fiware components:

PostgreSQL, MySQL, MongoDB, and Hadoop cluster.

To simplify the maintenance:

PostgreSQL will replace MySQL in each component where it’s possible, unless it’s not possible to change easily to PostgreSQL, [MariaDB](https://mariadb.org/) should replace MySQL without any change.

To rationalize the architecture, each kind of database service will be deployed once in the platform, shared by the client components.

Each of the database service will be based on a cluster:

* This is native architecture for MongoDB
* PostgreSQL databases will be based on a Citus database cluster
* MySQL/MariaDB databases will be based on a Galera database cluster

May be thinking to master data management at the occasion of this project is an opportunity to consider

In particular, to reach GDPR conformity more easily, by using appropriate tools to manage data and the access rights.

#### Transaction volumes

Write transactions volume:

In the excel document about data input traffic, the number of transactions per day for data acquisition has been evaluated around 620 000, this is a very low traffic.

A data acquisition transaction consists in the transformation of the message to NGSI, the update of the data in real time storage and insert it in history data store.

the top peak known demand is for tramway data (216 data/28 trains/every minute), for existing T1 line,

even we project to 100 trains with the 3 lines working, it’s less than 2 transactions per seconds.

Read transaction volume:

To give a global evaluation at this step of the project is not easy, since it depends mostly on the type of application, and the traffic they should generate

The database clusters will be easily horizontally scalable, to support read transaction increase in case of need.

#### Storage Sizing

Depending on the component, some storage volumes have to be used, to persist data outside of the container itself.

For most of components except database servers, the storage needs are limited to the container image footprint (from 40MB for simple to 1GB for very complex images), and some configuration files (important for the container, negligible in size).

In section 3, for each component indication about the storage requirements will be assessed.

For database clusters, the problematic is different and is described in section 3 as well.

#### Concurrent user & devices

The scalability of the platform using containers, allows to virtually support any numbers of concurrent users and devices, by deploying more instances of services, providing the provisioning of enough computing nodes as virtual machines or bare metal servers to absorb the load.

By starting with a simple configuration on 3 or 4 nodes, we will evaluate performance through some simple use cases scenarios, with a monitoring system beside, which store history.

#### Data import, export and models

Data existing in the current data warehouse will be imported into the new platform,

The main concern remains the current data model adaptation to existing [Fiware data models](https://www.fiware.org/data-models/) and eventually propose amendments when needed.

This is currently under study, in order to be able to propose a table of correspondence between the existing data source attributes and the attributes defined in the schema for the destination data object (an NGSI entity), that, if possible, match an existing Fiware data model.

In case new attributes need to be added to a Fiware proposed model, or new data model, the proposal should be sent to Fiware for validation, respecting the given guidelines on [How To Contribute](https://github.com/fiware/dataModels#how-to-contribute).

#### Disaster recovery

The 3 data centers should be involved in the smart city platform, then in case of serious problem in the main and secondary data center, the third DC, more protected from disaster like flood, can take over, and keep the platform online.

The databases used within the platform, must be replicated in real time on the third data center, to be able to use it in the shortest time possible.

#### GDPR

Observing General Data Protection Regulation will be mandatory on May 25th 2018, it’s a recent European directive defining new rules for personal data management.

Here are the major changes that are mentioned in this new legislation:

* **Consent:** Consent of personal data must be freely given, specific, informed and unambiguous. Consent is not freely given if a person is unable to freely refuse consent without detriment.
* **Accountability and privacy by default:** The GDPR has placed great emphasis on the accountability for **data controllers** to demonstrate data compliance. They will be required to maintain certain documentation, conduct impact assessment reports for riskier processing and employ data protection practices by default, such as data minimization.
* **Notification of a data breach: Data controllers** must notify the Data Protection Authorities as quickly as possible, where applicable within 72 hours of the data breach discovery.
* **Sanctions:** This new legislation allows the Data Protection Authorities to impose higher fines – up to 4% of annual worldwide turnover. The maximum fines can be applied for discrepancies related to international data transfers or breach of processing principles, such as conditions for consent. Other violations can be fined up to 2% of annual worldwide turnover.
* **Role of data processors: Data processors** will now have direct obligations to implement technical and organisation measures to ensure data protection, this could include appointing a **Data Protection Officer** if needed.
* **One stop shop:** This legislation will be applicable in all EU states without the need of implementing national legislation. Having a single set of rules will benefit businesses as they will not need to comply with multiple authorities, streamlining the process and saving an estimate of €2.3 billion a year.
* **Removal of notification requirement: T**he requirement of notifying or seeking approval from a Data Protection Authority is going to be removed in many circumstances. This decision is made to save funds and time. Instead of notification the new directive requires **data controllers** to put in place appropriate practices for large scale processing in the form of new technology.
* **Right to be forgotten:** Persons will be able to require their data to be deleted when there is no legitimate reason for an organization to retain it. Following, the organisation must also take appropriate steps to inform any third party that might have any links or copies of the data and request them to delete it.
* **Expanded territorial reach:** Companies that are based outside of the EU, but targeting customers that are in the EU will be subject to the GDPR which is not the case now. (Not Applicable for the project).

Data Protection Act identifies Data controllers and Data Processors as separate jobs:

<https://ico.org.uk/media/for-organisations/documents/1546/data-controllers-and-data-processors-dp-guidance.pdf>

The new GDPR rules described above will have to be applied mostly in the applications involving the use of personal data (in particular data from citizens).

However, as the platform will be recipient of many different kind of data, it could be hard to embrace the complexity of the data sets possible entanglement, some data can be related to citizens, because coming from objects/sensors at home, even weared, or simply published through applications.

Then to have a Data Protection office is worth to consider (if not already acted), but this is beyond the scope of this document.

The multitenant nature of the Fiware platform help a lot to data protection & privacy by dividing into independent services with secure and controlled (authorized and authenticated) access to data-sets, provided by global security IDM component.

However, in the case of cross usage of multiple history datasets like in some data-science cases, data must be anonymized and minimized, before being delivered for analysis, this is to consider either at data acquisition step and data consuming step.

## System Architecture Component Definitions

### Docker Images Management Components

MNCA needs to manage its own Docker images in a local Docker Images Registry, [Portus](http://port.us.org/) has been chosen as the component for this role.

It’s not a Fiware functional module, but an infrastructure element to manage a Docker registry (Docker images storage).

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql or MySQL (default) => Citus/PostgreSQL |
| Application | Docker Registry  NGINXor[**Puma**](http://puma.io/)(Ruby web server)  Portus web UI (developed in Ruby) |

Portus provides mainly authentication towards a Docker registry, which have to be started in a container: [Docker documentation](https://docs.docker.com/registry/deploying/#copy-an-image-from-docker-hub-to-your-registry).

Portus and the docker registry will be deployed as services, through a Docker compose or stack file: [example of secure compose file](https://github.com/SUSE/Portus/blob/master/examples/compose/docker-compose.yml)

There’s no real need of scalability for this component, only fault tolerance and high availability are required.

It should be installed on each Swarm manager node.

### Docker Swarm Management Component

Swarm manager is the orchestrator of the services based on the components of the platform.

It’s part of Docker software suite.

Swarm is cluster management integrated with Docker Engine (appeared in version 1.12)

To insure high availability several Docker nodes should be swarm manager, at the minimum one in each of the 3 DCs, there’s no real need of scalability, since it’s an internal orchestration management interface.

Even the choice is to setup the platform using Docker Swarm via Docker CLI, it could be valuable to use a container management solution such as [Rancher](http://rancher.com/)

Rancher is a complete container management open-source platform that makes managing and using containers in production really easy.

It allows to deploy services stacks over several Docker orchestration tools such as: Kubernetes, Swarm, Mesos.

It allows adding hosts directly from cloud providers or adding a host that’s already been provisioned. For cloud providers, host are provisioned using docker-machine and support any images that docker-machine supports.

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | MySQL (default) |
| Applications | Rancher server  Rancher UI  all in one Docker container |

### Security Component IDM GE

This component is the Fiware GE responsible for Fiware platform global user identity & access rights management

It is composed of 3 elements:

* the Identity Manager GE, in fiware catalog: [Keyrock](https://catalogue.fiware.org/enablers/identity-management-keyrock)
* the Access Control GE, in fiware catalog: [AuthZForce](https://catalogue.fiware.org/enablers/authorization-pdp-authzforce) (works with PEP Proxy)
* GE module to add security to backend applications, in fiware catalog: [PEP Proxy](https://catalogue.fiware.org/enablers/pep-proxy-wilma)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | changed to Postgresql/Citus cluster instead of MySQL |
| Applications | Keyrock (Keystone API backend + front-end: Horizon Django based application)  AuthZforce  PEP Proxy |

The security component is central to the platform, and so, used by most of other components from the platform to provide them authentication and authorization management, such as Orion Context Broker, Wirecloud, CKAN, CEP, Spagobi/Knowage, as well as Business ecosystem.

The Keystone IDM, back-end of Keyrock, can easily connect to an external LDAP, to authenticate and authorize users registered in it, new users can still be added and managed through Keyrock, but not registered in the LDAP (read only), once a user has been identified through LDAP, the access rights are managed through the security component.

AuthZForce is the PDP ((Policy Decision Point), PEP Proxy the PEP (Policy Enforcement Point).

Applications will be deployed through a service (in a Docker Stack file) providing high-availability and easy scalability, and will use the PostgreSQL/Citus cluster for data persistence.

### Orion Context Broker Component

This component is central to the Fiware platform: this is the data exchange bus of the platform.

The reference implementation of the Publish/Subscribe Context Broker GE.

Orion Context Broker allows to manage the entire lifecycle of context information including updates, queries, registrations and subscriptions.

It is an [NGSIv2](http://fiware.github.io/context.Orion/api/v2/stable/) server implementation to manage context information and its availability.

On Fiware catalog: [Orion Context Broker](https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker)

On Readthedocs: [Orion documentation](https://fiware-orion.readthedocs.io/en/master/)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | MongoDB |
| Applications | Orion context broker  Rush notification relayer (queue notifications to reduce load on Orion) |

Context information follows the model presented in figure 5:

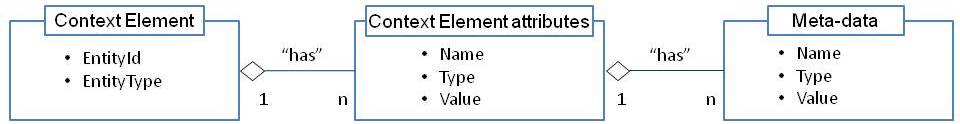


Figure 6: Context Element Model

A context element must provide an Entity ID, and an Entity Type, it can have as many attributes as needed, and each attribute can have as many Meta-data as needed.

The NGSI RESTful API manipulate context elements in JSON format.

Context Broker as a central component interact with many other components such as IOT Agents, Cosmos (Cygnus and Comet), CEP, Wirecloud

Context Broker is secured through the Keyrock IDM security component and PEP proxy.

To provide high availability and scalability, the Orion component is deployed as a service through a Docker stack file, meaning that several container instances of Orion will run to provide HA, scalability.

To reduce the load on Orion for managing notifications it’s possible to use a specific additional component: [Rush](https://fiware-orion.readthedocs.io/en/master/admin/rush/index.html) as a notification relayer ([Rush on Telefonica github](https://github.com/telefonicaid/Rush)), a Docker file: [OrionRush on github](https://github.com/Geonovum/sospilot/tree/master/src/fiware/docker/orionrush)

### MongoDB Cluster Component

This component is the [MongoDB](https://www.mongodb.com/) NoSQL database cluster global to the platform.

Several Fiware components use MongoDB to persist data, among them: Orion, IDAS, Comet STH.

Clustering is native in MongoDB, a MongoDB cluster can be scaled horizontally, sharding can be done on documents through replicas.

The component is deployed as a cluster of MongoDB nodes as containers in the swarm, as a Docker service through a Stack file.

Each container should point to a volume on the host for data persistence.

### Citus/PostgreSQL Cluster Component

This component is the cluster of [PostgreSQL](https://www.postgresql.org/) database servers of the Fiware platform.

The component is based on [Citus](https://www.citusdata.com/product/community), an open source clustering solution for PostgreSQL.

PostgreSQL is used by several components in the platform: such as CKAN, Wirecloud, and those that will be changed from MySQL to PostgreSQL like Keyrock, Biz Eco System.

Citus extends PostgreSQL to support distributed SQL queries. On top of PostgreSQL, Citus comes with its own transparent sharding, replication, distributed query planner and executor logic which enable execution of distributed SQL queries in parallel. This provides Hadoop-like fault tolerance, scalability and recovery from mid-query failures while allowing large datasets to be queried orders of magnitude faster than what has been possible on PostgreSQL before.

In Citus architecture, the manager is a specific PostgreSQL database server instance that needs a replication to insure manager fail-over and provides complete high availability.

Several replication solutions exist for PostgreSQL, the recommendation from CitusData is [PostgreSQL integrated Streaming replication](https://www.postgresql.org/docs/9.6/static/warm-standby.html#STREAMING-REPLICATION).

### Galera/MySQL Cluster Component

This component should be the cluster of MySQL database servers of the Fiware platform.

This component will be installed only if changing from mySQL to PostgreSQL for a module is too complex or impossible.

Any MySQL database that can’t be replaced with PostgreSQL, should be replaced by MariaDB,

To provide high availability and scalability the component will be based on a MySQL/MariaDB clustering solution: [Galera cluster](http://galeracluster.com/products/).

There is a MariaDB and Galera integration chapter on [MariaDB web site](https://mariadb.com/kb/en/library/galera-cluster/).

### Cosmos Big Data Component

This component is the Big Data and data history GE of the Fiware platform.

As the most complex component, it’s composed of many modules:

* Cygnus, as an interface between the context broker and the different data history storages
* Comet STH for Short Term History storage, uses MongoDB to persist the data sent through the context broker.
* An Hadoop ecosystem (HAAS) to store and process (batch or stream) long term history for analytics purposes.

Hadoop HDFS provides Distributed storage.

Distributed batch processing (with MapReduce).

Distributed stream processing through Sinfonier (Apache Storm based) or Spark.

Also, almost any software of the Hadoop ecosystem can be used in Cosmos: such as Hive, Pig, zookeeper, yarn, HBase…

On Fiware catalog: [Cosmos](https://catalogue.fiware.org/enablers/bigdata-analysis-cosmos)

On readthedocs: [Cosmos documentation](http://fiware-cosmos.readthedocs.io/en/latest/)

On Fiware Forge: [Architecture documentation](http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.BigData)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | MongoDB cluster |
| Applications | Cygnus  Comet STH  Hadoop ecosystem: Sahara HAAS engine or HAAS light version engine  Tidoop (MapReduce)  Sinfonier (stream analytics Apache storm based) |

### Knowage Analytics Component

This component is for business intelligence, Fiware have originally chosen SpagoBI as the reference platform for this Analytics GE.

In the Fiware catalog is proposed the evolution of SpagoBI to a more integrated suite with all modules included Knowage

On Fiware catalog: [Knowage](https://catalogue.fiware.org/enablers/data-visualization-knowage)

On readthedocs: [Knowage in Fiware](http://knowage.readthedocs.io/en/latest/)

Knowage is a professional open source suite for modern business analytics over traditional sources and big data systems.

This GE provides the following features:

* BD (big data), to analyse data stored on big data clusters or NoSQL databases
* SI (smart intelligence), the usual business intelligence on structured data, but more oriented to self-service capabilities and agile prototyping
* ER (enterprise reporting), to produce and distribute static reports
* LI (location intelligence), to relate business data with spatial or geographical information
* PM (performance management), to manage KPIs and organize scorecards
* PA (predictive analysis), for more advanced analyses
* EI (embedded intelligence), to link Knowage with external solutions provided by the customer or third parties.

Knowage can connect to CKAN and show the catalog of data sets in it, allows to search in it, in order to be able to use it (only CSV & XLS data source for the moment, other formats will be added later).

Knowage can also define an NGSI dataset (query data from the Orion Context Broker

It will be configured to use the Keyrock IDM component for authentication.

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | PostgreSQL/Citus for metadata storage |
| Applications | A set of java modules deployed under a Tomcat |

The composer part of Knowage, shouldn’t be exposed to public use, if some dashboards or reports visualization needs to be published, it need to be integrated in applications in an iframe component or by using the API to get the Knowage generated content.

### CKAN Open Data Component

This component is the chosen Fiware GE to manage Open Data.

CKAN GE at Fiware: <https://catalogue.fiware.org/enablers/ckan>

Fiware Docker installation: <https://github.com/okfn/docker-fiware-ckan>

Official installation guide : <http://docs.ckan.org/en/ckan-2.3.5/maintaining/installing/install-using-docker.html>

The CKAN GE provides the following features:

* Complete catalog system with easy to use web interface and a powerful API
* Strong integration with third-party CMS’s like **Drupal** (currently in use at MNCA) and WordPress
* Data visualization and analytics
* Workflow support lets departments or groups manage their own data publishing
* Fine-grained access control
* Integrated data storage and full data API
* Federated structure: easily set up new instances with common search

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus cluster  Apache Solr (for search feature) |
| Applications | Api and WEB UI |

### IDAS IOT Component

The IDAS component is an implementation of the Backend Device Management GE.

It provides software for IOT use cases, like:

* to manage communication between sensors & actuators/devices/gateways and the data context broker using NGSI REST API.
* to manage provisioning of devices through a REST API, it is composed of several components organized by south bound protocols (UL2, JSON, LWM2M, Sigfox)

On Fiware Catalog: [IDAS](https://catalogue.fiware.org/enablers/backend-device-management-idas)

Documentation on the Fiware IOT stack : [by Telefonica](https://fiware-iot-stack.readthedocs.io/en/latest/)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | MongoDB |
| Applications | [UL2.0 IOT Agent](https://github.com/telefonicaid/iotagent-ul)  [JSON IOT Agent](https://github.com/telefonicaid/iotagent-json)  [OMA LWM2M IOT Agent](https://github.com/telefonicaid/lightweightm2m-iotagent)  [SIGFOX IOT Agent](https://github.com/telefonicaid/sigfox-iotagent)  [IOT Agent library to develop other agents](https://github.com/telefonicaid/iotagent-node-lib)  IOT Agent Manager (under upgrade for R5 release)  IoT Edge Manager (forthcoming for R5)  [IOT Orchestrator with portal UI](https://github.com/telefonicaid/orchestrator) (from Telefonica)  MQTT broker like [Mosquitto](http://mosquitto.org/) is needed for MQTT connectivity of IOT agents. |

As the IOT agent components provide only a REST API to interact with, it lacks a web user interface for device management in the current Fiware catalog.

The [IOT Agent Manager](https://github.com/telefonicaid/iotagent-manager) is under porting in Nodejs for release 5, it should provide the device management layer.

Most of the IOT modules in Fiware catalog come from [Telefonica, in their github](https://github.com/telefonicaid), they provide a component called [Orchestrator](https://github.com/telefonicaid/orchestrator), that allows to interact with the different modules: IOT agents, perseo CEP, Orion, IDM, this orchestrator component also provide a web portal ready to use, but not open-source.

An IOT agent for [OPC UA](https://en.wikipedia.org/wiki/OPC_Unified_Architecture) is under development: <https://github.com/BEinCPPS/idas-opcua-agent> it should allow to exchange information between an OPC server and an NGSI context broker.

After analysis of already used data sources, it appears that none is immediately compatible with the proposed IOT agents, then some adapters need to be developed, to be able to store in Fiware, the data coming from those different data sources.

The existing data sources are most of time files in different format (XML, CSV, formatted text), coming through FTP, or got through HTTP

Those data should be converted to JSON in a first step, and posted to an MQTT broker, then a **JSON IOT Agent** subscriber to the topic on the MQTT broker, should convert into NGSI and post to the context broker.

The first step (data acquisition, conversion to JSON and publish to MQTT) will be done through specific programs or scripts, one per type of data source.

The components will be deployed as docker services through stack files.

### CEP Component

The CEP component is the Complex Event Processing GE from the Fiware platform.

In Fiware catalog is proposed an implementation from IBM of the CEP module: [PROTON](https://catalogue.fiware.org/enablers/complex-event-processing-cep-proactive-technology-online)

It is a scalable integrated platform to support the development, deployment, and maintenance of event-driven applications.

To summarize: through the CEP component it’s possible to configure rules on incoming data (like: if a fridge temperature is higher than 4°C),

that if verified, generate an event/action (an alarm is sent, mail, SMS, or displayed on a cockpit/dashboard)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | None in this version |
| Applications | Proton is based on Tomcat and uses 4 wars:  ProtonOnWebServerAdmin  ProtonOnWebServer  AuthoringTool  AuthoringToolWebServer |

The CEP Proton component will be deployed as a Docker service, using stack file.

### Application Mashup Components

Wirecloud is the Fiware GE to build Rich Internet Applications (RIA), using semantic technologies to offer a next-generation end-user centered web application mashup platform.

On Fiware Catalog: [Wirecloud](https://catalogue.fiware.org/enablers/application-mashup-wirecloud)

On Readthedocs : [Wirecloud Documentation](https://wirecloud.readthedocs.io/en/stable/)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus |
| Applications | Web application  back-end server |

Wirecloud is a web application mashup platform aimed at allowing end users without programming skills to easily create web applications and dashboards/cockpits.

It integrates completely to Fiware architecture, by connecting directly to Context Broker and historic data storage components such as Comet STH.

Mashups and Dashboards made with Wirecloud, can be monetized through the business ecosystem component

Alternatively a small javascript library like [Freeboard](https://github.com/Freeboard/freeboard) can be used to visualize data coming from NGSI context broker: [news on Fiware](https://www.fiware.org/2015/07/13/fiware-orion-data-source-now-available-for-freeboard/). (an [Orion FIWARE datasource](https://github.com/telefonicaid/fiware-connectors/tree/master/FreeBoard-Orion-Plugin) has been developed by Telefonica)

Freeboard can be used also from web where you can publish dashboards: [Freeboard.io](https://freeboard.io/)

It can also be deployed on a web server, on the platform, but there’s no security or publishing management provided, it has to be added, by passing through a proxy.

As it’s pure javascript library and some json definition and configuration, it can be integrated in a web application that is proposed in the market place.

### Business Ecosystem Component

The Business API Ecosystem exposes its complete functionality through TM Forum standard APIs; concretely, it includes the following key features:

* Support for the management of catalogs, products, and offering
* Support for rich pricing models, including recurring payments, pay-per-use, etc.
* Support for accounting callbacks
* Support for billing and charging
* Integrated support for PayPal, including customer charges and seller payments
* Support for revenue sharing, including models with multiple stakeholders involved

Link to Fiware catalog: [Biz Api Ecosystem](https://catalogue.fiware.org/enablers/business-api-ecosystem-biz-ecosystem-ri)

Link to the documentation: <http://business-api-ecosystem.readthedocs.io/en/latest/>

This component is the Market Place of the Smart City platform, it allows to leverage:

Authorization access to services, and their monetization, even with several stakeholders involved that share revenues,

services here includes the applications available in the catalog, the data sets in CKAN, the available APIs.

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus (change from MySQL) |
| Applications | Web application  back-end servers:   * TM Forum APIs reference implementation (catalog, ordering, inventory, usage, billing, party Apis) * Charging Backend, * Revenue Sharing Service (RSS) * Logic Proxy |

### Development Components

Fiware was proposing to use [FusionForge](https://fusionforge.org/) (the software suite originated from Forge that was running SourceForge.net), Eclipse as an IDE as the set of tools for development.

Now with the advent of Docker, Eclipse have evolved towards the container paradigm and propose [Eclipse Che](https://www.eclipse.org/che/) as an IDE & Developer Workspace server that works with Docker.

It allows to develop Docker ready applications, in addition it provides a RESTful API to interact with.

If for many applications like dashboards, using Wirecloud component is a good solution, for more complex or even mobile application it is not adapted, so that Eclipse Che IDE is an appropriate solution to develop Docker ready applications.

Eclipse Che is the next generation of IDE, provides integrated connectors to git and SVN, so that it can be easily connected to the Gitlab of MNCA.

An Eclipse Che server can be set in a multi-users environment, in standard installation it uses Keycloack as IDM provider, and a PostgreSQL database for metadata.

As Keycloak can replace Keystone as an authentication service (OAuth provider), in OpenStack deployment, and Eclipse Che can also use OAuth from GitLab or WSO2, if needed Keyrock can be also used as the OAuth provider giving SSO to developers using the Eclipse Che server.

Keycloak comes also with a built-in LDAP/AD plugin, which supports password validation via LDAP/AD protocols and different user metadata synchronization modes, so that internal users already registered on MNCA ADAM, can be identified in Eclipse Che.

An Eclipse Che Server using the IDM component, is a good candidate for the Paas IDE for IOE of the MNCA smart city platform.

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus |
| Applications | Elipse Che |

### Monitoring Components

[Portainer](https://portainer.io/) will be used for real time management & monitoring of services and the composing containers of the swarm.

Portainer on [Github](https://github.com/portainer/portainer/blob/develop/README.md).

Portainer will be replicated on several nodes with several instances, ideally on each swarm manager (mandatory to have vision of the swarm).

As Portainer don’t use a Database, but store it’s configuration in files, mainly for user and dependencies management, so that those configuration files should be stored in a secured NFS storage volume, shared by all instances.

[Centreon](https://www.centreon.com/) is already in use for monitoring at MNCA.

containers should be instrumented with needed check modules to monitor services and nodes in the platform from the Centreon, an additional Centreon service could be added in the platform if needed.

One thing to think about is the expansion of the platform, in terms of numbers of services deployed, this will lead to complexify the task of monitoring configuration if it remains manual, so that it’s worth to think about automatic services discovery to simplify and automate monitoring for OPS

One solution should be to use [Consul](https://www.consul.io/) from Hashicorp for services discovery, but Centreon provides also a module for [auto discovery of services](https://documentation.centreon.com/docs/centreon-auto-discovery/en/latest/) (commercial) or a community version: [centreon-discovery](https://github.com/Centreon-Community/centreon-discovery)

If Rancher is adopted it provides monitoring on containers and nodes, and in swarm mode, it deploys also a Portainer.

### Poi Data Provider Component

Fiware GE to provide Point Of Interest (POI) management.

On Fiware Catalog: [POI Data Provider](https://catalogue.fiware.org/enablers/poi-data-provider)

POI (Points of interest) Generic Enabler is a web server kit that supports

* storing information related to locations
* serving queries by location and other criteria
* can be configured to meet your data needs

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus + PostGIS  MongoDB |
| Applications | API back-end server  PHP web application |

### GIS Data Provider Component

Fiware GE to provide spatial data storage.

This GE is able to host geographical data and serve it in 3D form (where applicaple) to both mobile and web clients. The GE implementation is based on open source Geoserver project (GPL licenced) and W3DS (Web 3D Service) extension.

On Fiware catalog: [GIS data Provider](https://catalogue.fiware.org/enablers/gis-data-provider-geoserver3d)

On readthedocs: [GIS Data Provider](http://gisdataprovider.readthedocs.io/en/latest/index.html)

|  |  |
| --- | --- |
| **Architecture Component** | **Component Elements** |
| Database Server | Postgresql/Citus + PostGIS |
| Applications | GeoServer |

# Section 3 SYSTEM ARCHITECTURE DESIGN

*The* ***System Architecture Design*** *section provides detailed descriptions of each product implementing architecture components, and explains the rationale for product selection.*

*For each* ***System Architecture Component*** *(identified in Section 2.3 above), this section describes: specific* ***Component Functions****, requirements and other* ***Technical Considerations*** *that were used in the decision-making process, as well as any specific* ***Products*** *selected to implement this component. The* ***Selection Rationale*** *identifies any other products that may have been considered, and provides rationale for the decision.* ***Architecture Risks*** *identifies any potential risks associated with the architecture element.*

## Docker Images Management Component

### Component Functions

This component is not a Fiware functional component, it is used to manage the access to Docker Images used to deploy the smart city platform in production, in a Docker Images Registry, generally private.

It is a manager over an installed Docker registry, that is used mainly to authenticate users to authorize them to manipulate Docker Images files from the attached Registry.

A Docker registry is a repository of built Docker images of containers (DockeHub is a public Docker registry).

Registry management is needed in production, but also in development and QA steps of a project

If Portus is the choice for production container images management,

GitLab is the container images management to use while development and QA.

### Technical Considerations

Portus an [open source software originated from OpenSuse](https://github.com/SUSE/Portus), written in Ruby.

It provides its own security access layer that can be configured to use an LDAP server for the authentication, then it is possible to configure the component to use MNCA MS ADAM identity management.

At least one local Docker registry must be installed as a Docker container.

Portus component needs a database, MySQL is proposed by default, PostgreSQL is also supported.

The database is needed for user management.

Portus can be configured to use an LDAP server for the authentication: [Portus LDAP support](http://port.us.org/features/2_LDAP-support.html), so it can be interfaced with the MNCA ADAM.

Portus and the Docker registry will be deployed as services, through a Docker compose or stack file: [example of secure compose file](https://github.com/SUSE/Portus/blob/master/examples/compose/docker-compose.yml)

Incoming port: 3000/TCP

### Selected Product(s)

The selected product to manage images in docker registry is Portus, but the registry itself is provided through a Docker registry container, so that Portus mainly provides authentication to allow access to the images

GitLab provides also Docker registry management for container images: [GitLab Container Registry](http://docs.gitlab.com/ce/user/project/container_registry.html).

It allows the management of Docker container images in Gitlab this could be used in development and QA (testing), and keep Portus and the associated registry as the registry for production.

### Selection Rationale

Several alternatives exist for Docker Images management UI, Portus is a well-known open-source distribution provided by [OpenSuse](https://www.opensuse.org/).

Portus has already been adopted during MNCA Fiware test by Olivier Sevilla, summer 2017.

Nevertheless, for development and QA, the use of specific registry attached behind the Gitlab of MNCA will allow to manage lifecycle of the Docker images, before releasing it in production.

GitLab like Portus is not a repository itself, it is a registry manager,

### Architecture Risks

The MNCA Gitlab should receive the Docker images in its own registry for development and QA ([just need to [install if not] activate it](https://docs.gitlab.com/ee/administration/container_registry.html#enable-the-container-registry))

The registry

## Docker Swarm Manager Component

### Component Functions

This is not a real component, it’s an artifact included in Docker (since 1.12 version), which allows to manage a Docker Swarm.

The main features are initialize a Swarm, add a machine/node to a swarm, remove a node from a swarm, renew certificates

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## Security Component IDM GE

### Component Functions

Manage globally the security for the Fiware platform, in terms of identity and access authorization to services and data.

The Keyrock module provides identity management Keystone back-end, the integrated web UI is provided through Horizon front-end, which allows user accounts management.

AuthZForce is the PDP ((Policy Decision Point), PEP Proxy the PEP (Policy Enforcement Point).

The AuthZforce component provides roles and access rights management based on XACML

The PEP Proxy components provides Keyrock security to others back’end of the platform such as Orion context broker.

The 3 components together offer 3 levels of security:

* Level 1: Authentication
* Level 2: Basic Authorization
* Level 3: Advanced Authorization

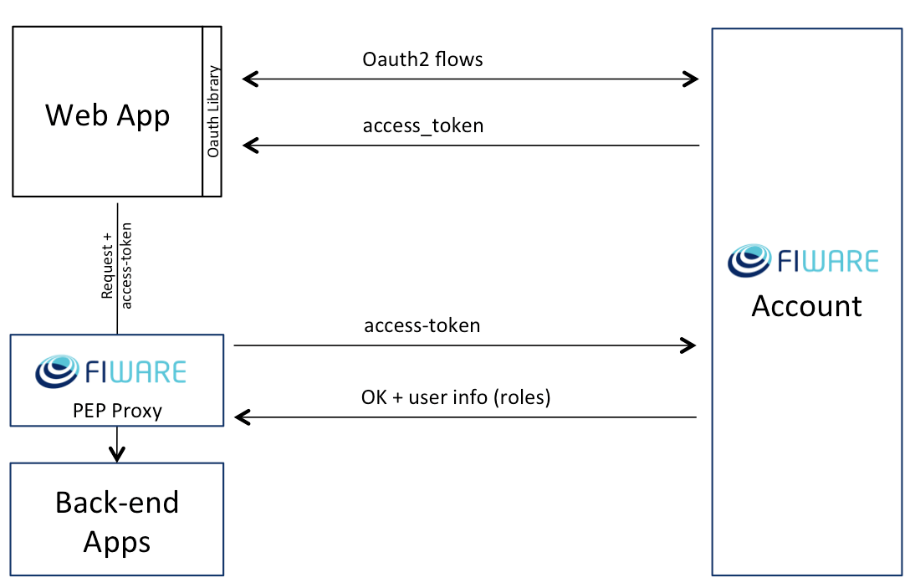


Figure 7: Level 1: Authentication only (Fiware Account is Keyrock IDM)

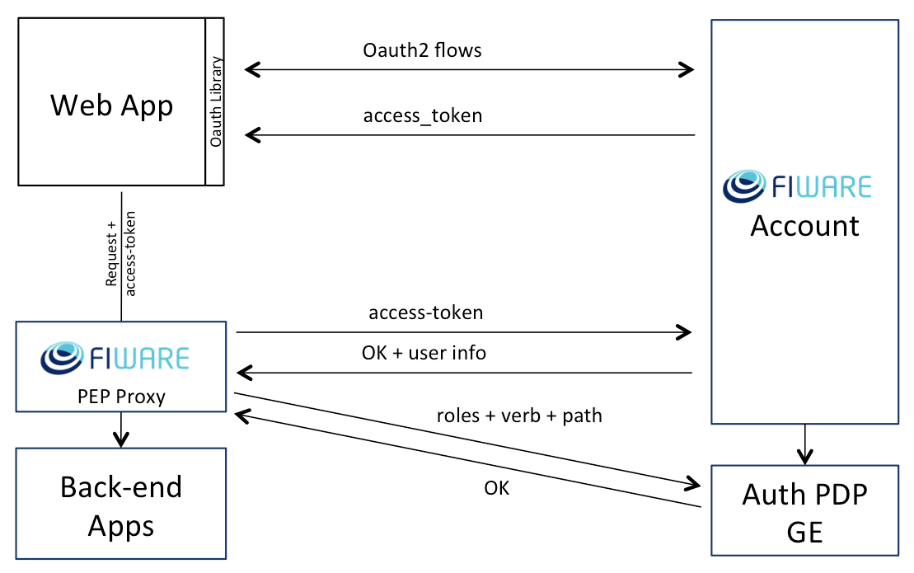


Figure 8: Level 2: Authentication & Basic Authorization

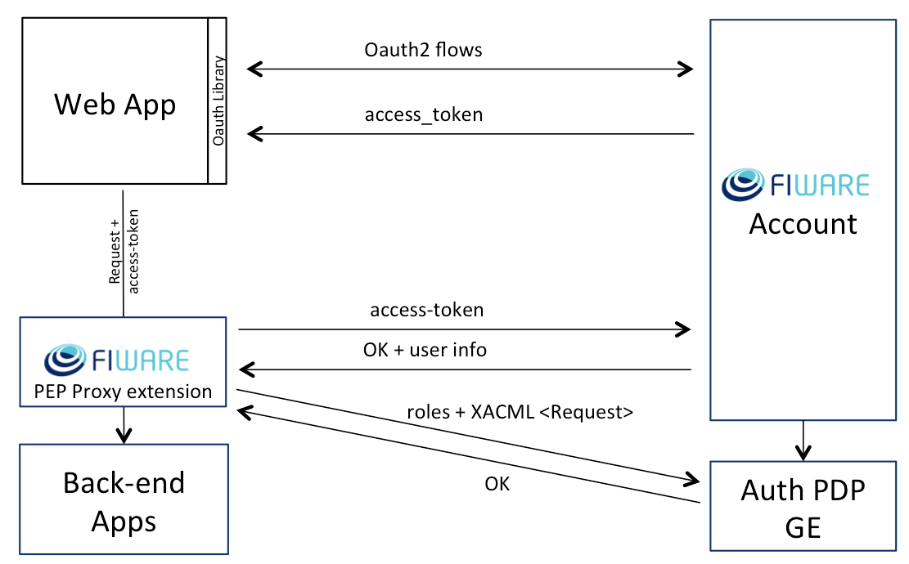


Figure 9: Level 3: Authentication & Advanced Authorization (using XACML).

### Technical Considerations

It’s based on **Keystone**, the IDM component of OpenStack cloud software suite.

it uses the standard OAuth2 protocol

it can connect with an LDAP: [Integrate Identity with LDAP](https://docs.openstack.org/keystone/pike/admin/identity-integrate-with-ldap.html), the LDAP can be the MNCA ADAM: [Keystone with ADAM](https://wiki.openstack.org/wiki/HowtoIntegrateKeystonewithAD), just matter of configuration.

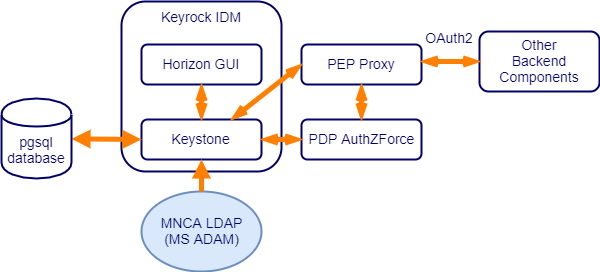


Figure 10: IDM components organization

Incoming ports:

* Keyrock API endpoint: 5000/TCP
* Horizon UI: 8000/TCP (in production 80 through NGINX)
* Keystone Admin endpoint: 35357/TCP
* PDP Authzforce:
  + 8080/TCP
  + 22/TCP

### Selected Product(s)

Keyrock identity manager is based on OpenStack **Keystone**

**Keystone** provides authentication, authorization and service discovery mechanisms via HTTP primarily for use by projects in the OpenStack family.

It is most commonly deployed as an HTTP interface to existing identity systems, such as LDAP.

### Selection Rationale

Keyrock, PEP Proxy, PDP AuthZforce are the components proposed in Fiware catalog,

Keyrock is Based on OpenStack IDM component **Keystone**, which supports: Oauth2, OpenID connect, SAML and other protocols, OAuth2 is now widely used, Fiware mostly rely on it, but .

### Architecture Risks

As a globally used component for Identity, authentication and access rights management, it is a highly critical component.

The database must be maintained in a clustered environment that insure high availability as well as scalability, this is provided through the Citus cluster of PostgreSQL databases.

The involved application modules (Keyrock, AuthZforce, PEP Proxy), will be deployed as services through a docker stack file in the global swarm, it gives the possibility to specify the number of containers to run for each service, as the components are stateless, several containers can run together the same service, to provide High Availability and scalability.

## Orion Context Broker Component

### Component Functions

Manage context information exchange for the whole platform using NGSI/V2 Api, this is a core feature of the Fiware Platform, as it provides the link between the data coming from things in the field, where data models are many, and their usage in the applications.

In addition to the query/answer paradigm, the component provides a publish/subscribe usage paradigm

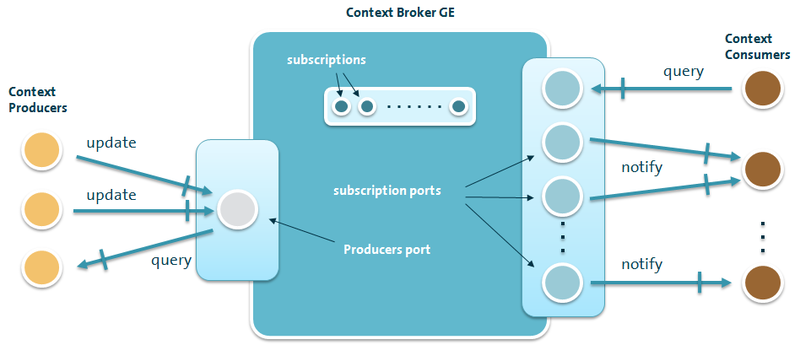


Figure 11: Context Broker interactions

### Technical Considerations

The Orion context broker is developed in C++

Incoming port: 1026/TCP

### Selected Product(s)

Orion Context broker is

There’s another implementation of an NGSI context broker originated from Orange:

But tailored for use at IOT gateway level, for example to manage a network of sensors relative to a specific activity among numerous smart city use cases

### Selection Rationale

Orion is the reference implementation of NGSI Context Broker in Fiware

### Architecture Risks

This component is central to the platform, because it’s the dorsal of the Fiware architecture

It can become a bottleneck in case of high traffic, so a particular attention is needed on response time, through specific monitoring.

The MongoDB database used by the context broker is a global MongoDB cluster, so that provide high availability and scalability.

The application component, Orion will be deployed as a service through a Docker stack file, to run several Orion containers, and will then provide high-availability and scalability.

you can also take advantage of Rush as notification relayer. Thus, instead of managing the notifications itself (including waiting for the HTTP timeout while the notification receives responses), Orion passes the notification to Rush, which in turn deals with it, to reduce the load on Orion.

[Rush](https://fiware-orion.readthedocs.io/en/master/admin/rush/index.html) as a notification relayer ([Rush on Telefonica github](https://github.com/telefonicaid/Rush)), a Docker file: [OrionRush on github](https://github.com/Geonovum/sospilot/tree/master/src/fiware/docker/orionrush)

To achieve best performance for the Orion context broker, we have to observe the tuning advices given in the [documentation on Readthedocs](http://fiware-orion.readthedocs.io/en/1.4.1/admin/perf_tuning/index.html)

## MongoDB Component

### Component Functions

MongoDB is a NOSQL document database, natively scalable by multiplying the nodes running the software and replicating the data upon the nodes

### Technical Considerations

Each MongoDB container will use a data directory on the host system (outside the container) and mounted to a directory visible from inside the container.

It is recommended to use MongoDB version 3.2 and higher like 3.4 (the latest stable as of today).

The standard MongoDB incoming port is 27017/TCP.

RAM memory sizing

Storage sizing

### Selected Product(s)

### Selection Rationale

Choice as database engine for several components proposed in Fiware.

A very popular scalable open source document oriented NoSQL database

### Architecture Risks

MongoDB will be deployed as cluster in the global platform Swarm, to achieve high availability several replicas will be deployed (each replica is a container), see MongoDB documentation

Even built from origin to scale, and fault tolerant, MongoDB is not optimized for analytics, that uses cross data queries, by using a solution based on PostgreSQL, like [Stampede](https://www.torodb.com/stampede/) for Mongo slaves, that could be mapped to the global PostgreSQL/Citus cluster ([cstore\_fdw](https://citusdata.github.io/cstore_fdw/)), high performance can be achieved on data reading and writing, while keeping long term history, then this is a solution to consider.

## Citus/PostgreSQL cluster Component

### Component Functions

Citus provide a way to make a cluster of PostgreSQL database servers, it will serve database(s) to any component of the platform that need it.

### Technical Considerations

### Selected Product(s)

### Selection Rationale

PostgreSQL is the open source database that is the closest to Oracle in term of performance in traditional transaction based SQL applications, this at no cost.

But PostreSQL have also introduced NoSQL capabilities, that allows them to challenge tenors like MongoDB in the domain.

It is easy to add spatial capabilities to Postgresql, with PostGIS, well known and very popular in the world of GIS.

Clustering solutions exist for PostgreSQL

### Architecture Risks

As the cluster which provides PostgreSQL services for all the platform, it’s a critical component.

To achieve best reliability and availability, the cluster should be deployed over the 3 MNCA data centers.

While not used most of the time, the nodes of the third data center should remain workers in the

## Galera/MySQL cluster Component

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## Cosmos Big Data Component with Cygnus & Comet

### Component Functions

Cygnus is a connector in charge of persisting certain sources of data in certain configured third-party storages, creating a historical view of such data.

The Short Time Historic (STH, aka. Comet) is a component of the FIWARE ecosystem in charge of managing (storing and retrieving) historical raw and aggregated time series information about the evolution in time of context data.

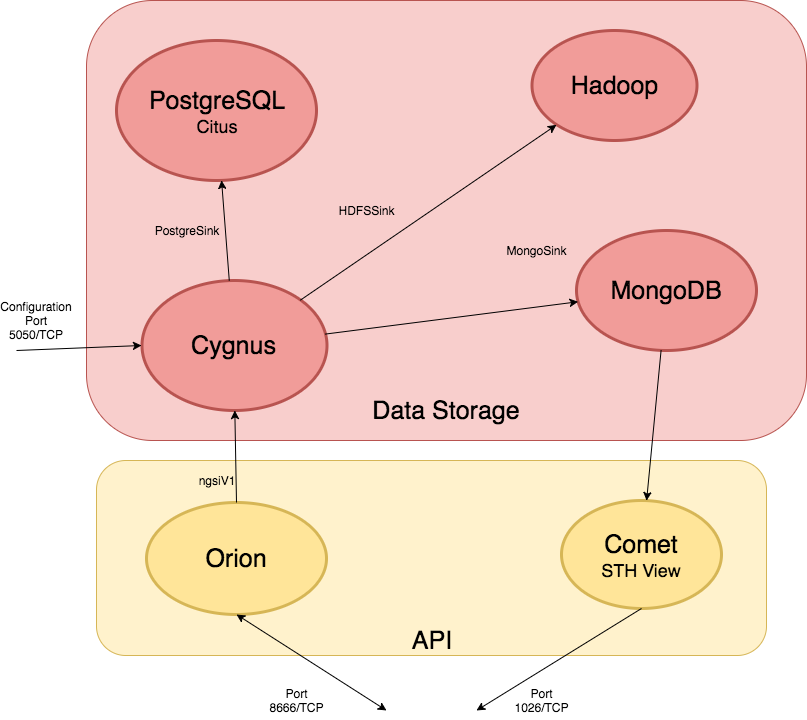


Figure 12 : Cygnus & Comet interactions

### Technical Considerations

Cygnus can be replicated to increase performance. It saves data from Orion with subscription and send data to a database like MongoDB or PostgreSQL.

Standard Cygnus port: 5050/TCP

Comet is an API to retrieve data from MongoDB were data are storage for STH (Short Time History ~ 1 year)

Standard Comet API port: 8666

### Selected Product(s)

Cygnus and Comet is the reference implementation of Big Data Component in Fiware

### Selection Rationale

### Architecture Risks

## Knowage Analytics Component

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

Knowage takes part in the FIWARE community, being the reference implementation of the Data Visualization GE.

### Architecture Risks

## CKAN Open Data Component

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## IDAS IOT Component

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## CEP Component

### Component Functions

CEP stand for Complex Event Processing

### Technical Considerations

### Selected Product(s)

After unsuccessful testing of Perseo, it has been decided to go with Proton CEP.

### Selection Rationale

In Fiware catalog is proposed an implementation from IBM of the CEP module: [PROTON](https://catalogue.fiware.org/enablers/complex-event-processing-cep-proactive-technology-online)

It is a scalable integrated platform to support the development, deployment, and maintenance of event-driven applications.

Telefonica propose also an implementation of the CEP GE: [Perseo](http://fiware-iot-stack.readthedocs.io/en/latest/cep/), as the CEP GE of their IOT stack

As well as Orange which propose [Cepheus](https://catalogue.fiware.org/enablers/iot-data-edge-consolidation-ge-cepheus)

It’s composed of a light NGSI broker and a CEP which features a dedicated REST management API to configure real time analysis on NGSI events.

The Perseo component has been tested by Faycal Boufares, and it appeared that it was not possible to run all components successfully (in particular Perseo Front-end is not working).

So, we’ll wisely use the Proton CEP proposed in the Fiware Catalog.

### Architecture Risks

## Application Mashup Components

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## Business Ecosystem Component

### Component Functions

The Business Ecosystem component is composed of several modules that cover the following features:

**TM Forum Apis:**

* Catalog management,
* Ordering management,
* Inventory management,
* Usage management,
* billing,
* customer,
* party APIs.

**Business Ecosystem Charging Backend:**

* Process the different pricing models, the accounting information, and the revenue sharing reports.
* Calculate amounts to be charged, charge customers, and pay sellers, using this above information.

**Business Ecosystem Revenue Sharing Service:**

Distributes revenues originated by the usage of a given service among the involved stakeholders.

It distributes part of the revenue generated by a service between the Business API Ecosystem instance provider and the Service Provider(s) responsible for the service, a service here refers to both final applications and backend application services.

**Business Ecosystem Logic Proxy:**

Is the endpoint for accessing the Business API Ecosystem:

* it orchestrates the APIs validating user requests, including authentication, authorization, and the content of the request from a business logic point of view.
* it serves a web portal that can be used to interact with the system.

This is the component that allows to monetize applications, APIs and also data, that are exposed through the smart city platform

### Technical Considerations

TM Forum APIs and Revenue Sharing Service requirements

* Java 8
* Glassfish 4.1
* MySQL 5.5 => to replace with PostgreSQL/Citus or use Galera Cluster and MariaDB if hard to change from MySQL to PostgreSQL

Charging Backend requirements

* Python 2.7
* MongoDB
* wkhtmltopdf

Logic Proxy requirements

* NodeJS 4.5.0+ (Including NPM)

### Selected Product(s)

The other components

### Selection Rationale

This is the reference component in Fiware Catalog,

The use of TM Forum APIs is a warranty of being able to conduct easily business relations with different skateh

### Architecture Risks

Is to be consider as a critical component, because it mediates the usage of all applications and their APIs.

## Development Components

### Component Functions

### Technical Considerations

### Selected Product(s)

### Selection Rationale

### Architecture Risks

## Monitoring Components

### Component Functions

Real Time monitoring with Portainer

Monitoring with history and notification through Centreon

### Technical Considerations

Centreon [Docker installation](https://github.com/julienmathis/centreon-docker)

### Selected Product(s)

### Selection Rationale

### Architecture Risks

# Section 4 System Construction Environment

*The* ***System Construction Environment*** *section details the various environments necessary to enable system construction and testing.*

*For each environment necessary for system construction (****Development, QA*** *and* ***Acceptance****), provide detailed specifications for the* ***Environment*** *and* ***Supporting Infrastructure*** *that will be used (including hardware*, network *and operating system requirements, all necessary installed packages and tools, and needed directory structures**that will be utilized to store all construction components).*

## Preliminary test Environment

A preliminary test environment has been setup:

* Authentication portal Horizon: **iot.testnca.org:8000**
* API d’authentification Keystone: **iot.testnca.org:5000**
* Sécurisation de l’API ORION via PEP-Proxy: **iot.testnca.org:8081**
* API Orion Context Brocker: **iot.testnca.org:1026**
* API IDAS http UltraLight: **iot.testnca.org:4041** *(Service/Device Creation)* **iot.testnca.org:7896** *(http UL2.0 for data update)*
* MongoDB: **iot.testnca.org:27017, iot.testnca.org:27018, iot.testnca.org:27019**
* CKAN Opendata Portal: **iot.testnca.org:8002**
* Wiki Documentaire Dokuwiki: **iot.testnca.org:8003**
* A portainer is also available: **iot.testnca.org:9000**
* A Docker registry has been installed: **registry.nicecotedazur.org:5000**

## Development Environment

### Developer Workstation Configuration

### Supporting Development Infrastructure Configuration

## QA Environment

### QA environment Configuration

### Supporting QA Infrastructure Configuration

## Acceptance Environment

### Acceptance environment Configuration

### Supporting Acceptance Infrastructure Configuration